数据科学基础
(With Python)

数据预处理

数据的缺陷

- 不完整:缺少某些特征值,缺少特征
  - 原因:有些特征的值难以得到,有些数据是因为早先认为不重要或由于误解而未记录,数据采集设备发生故障
- 不正常:噪声,错误,异常值
  - 原因:数据采集设备故障,数据录入错误,数据传输错误,技术限制.
  - 异常值不一定是错误数据

预处理技术

- 数据分析之前必须预处理.
  - 数据预处理占任务的80%以上工作量.
- 数据预处理技术
  - 数据清洗:平滑噪声,修正错误,填充缺失值,识别异常值
  - 数据转换:标准化,聚合
  - 数据约简:数据聚合,属性选择,维度约简
    - 数据集变小且仍能产生几乎相同的分析结果

数据清洗

- 检测缺陷
  - 数据录入错误,数据失效, 不一致表示, 不一致编码,记录数据的设备故障,系统错误,数据使用不当, ...
- 数据转换:一旦发现数据缺陷,需要定义转换来纠正
- 缺陷检测和数据转换反复进行
  - 这个过程是易错,耗时的
  - 转换可能带来新的缺陷

缺失值

- 实例在一些特征上没有观测值
  - 表示不存在
  - 存在但未观测到
- 首先应对缺失值本身进行分析,弄清原因及后果
- 对缺失值的处理
  - 删除
  - 填充

缺失值的处理

- 删除有缺失值的实例
- 填入缺失值
  - 人工填入合适值
  - 填入常数:如"未知"或"0"
  - 填入特征的均值
  - 填入与实例同属一类的实例的均值
  - 填入最可能的值

检测缺失值(1)

- pandas使用浮点值np.nan表示缺失值,习惯称为NA
- pandas对象的isnull()和notnull()用于检测NA

```
In [2]: from numpy import nan as NA
In [3]: s = pd.Series(['apple','huawei',NA])
In [4]: s
Out[4]:
0     apple
1    huawei
2       NaN
```

检测缺失值(2)

```
In [5]: s.isnull()
Out[5]:
0    False
1    False
2     True
```

- None在pandas中也视为NA

```
In [6]: s[0] = None
In [7]: s.isnull()
Out[7]:
0     True
1    False
2     True
```

删除缺失值(1)

- pandas对象的dropna
  - 对Series对象,dropna返回仅由非NA值构成的Series

```
In [9]: data = pd.Series([1,NA,3,NA,5])
In [10]: data.dropna()
Out[10]:
0    1.0
2    3.0
4    5.0
```

  - 也可利用isnull和布尔索引实现：

```
data[data.notnull()]
```

删除缺失值(2)

- 对DF:可删去全NA的行或列,也可只要含有NA就删除

```
In [12]: df = pd.DataFrame([[1,2],[3,NA],[NA,NA]])
In [13]: df.dropna()
Out[13]:
     0    1
0  1.0  2.0

In [16]: df.dropna(how='all')
Out[16]:
     0    1
0  1.0  2.0
1  3.0  NaN
```

删除缺失值(3)

- 如果删除含有NA的列,用参数axis=1

```
In [18]: df.dropna(axis=1)
Out[18]:
Empty DataFrame
Columns: []
Index: [0, 1, 2]
```

- 保留含指定数量(thresh参数指定)观测值的行

```
In [19]: df.dropna(thresh=1)
Out[19]:
     0    1
0  1.0  2.0
1  3.0  NaN
```

填充缺失值(1)

- 用fillna填充缺失值

```
In [27]: df.fillna(0)    # 填入常数
Out[27]:
     0    1
0  1.0  2.0
1  3.0  0.0
2  0.0  0.0      # 大量相同值可能影响分析算法!

In [28]: df.fillna({0:0,1:0.1})
Out[28]:
     0    1
0  1.0  2.0
1  3.0  0.1
2  0.0  0.1
```

填充缺失值(2)

- 参数method='ffill'可实现插值
  - limit限制插值的个数

```
In [30]: df.fillna(method='ffill')
Out[30]:
     0    1
0  1.0  2.0
1  3.0  2.0
2  3.0  2.0

In [31]: df.fillna(method='ffill',limit=1)
Out[31]:
      0    1
0  1.0  2.0
1  3.0  2.0
2  3.0  NaN
```

填充缺失值(3)

- 填充均值

```
In [36]: s = pd.Series([1,NA,3,NA,8])

In [37]: s.fillna(s.mean())
Out[37]:
0    1.0
1    4.0
2    3.0
3    4.0
4    8.0
```

异常值

- 异常值会影响学习算法和模型,导致模型在预测时不能适应新数据.
- 异常值可能是有效值,对这种异常值的解决方法是删除或降低权重
- 异常值还可能代表另一种分布,影响正在研究的数据样本,所以必须删除
- 异常值还可能就是错误数据,对此最好的解决方法是删除,当作随机缺失值
- 还可用均值等来替换

识别异常值(1)

- EDA分析和箱线图常用来识别异常值
  - z分数绝对值大于3
  - 小于Q1 - 1.5IQR和大于Q3 + 1.5IQR的值

```
In [92]: df = pd.DataFrame(np.random.randn(1000,3))
In [93]: df.describe()
                 0            1            2
count  1000.000000  1000.000000  1000.000000
mean     -0.004496    -0.015524     0.037189
std       1.003414     1.000842     1.037048
min      -2.717053    -3.315613    -3.235131
25%      -0.688916    -0.692900    -0.675153
50%      -0.017609    -0.011644     0.016412
75%       0.669723     0.641029     0.744385
max       3.413147     3.676253     3.788141
```

识别异常值(2)

- 第2列中的异常值

```
In [94]: c2 = df[2]
In [95]: c2[np.abs(c2)>3]
Out[95]:
527    3.788141
738   -3.235131
```

- 存在异常值的所有行

```
In [96]: df[(np.abs(df)>3).any(1)]   # 1即axis=1
Out[96]:
             0         1         2
70   -0.185640  3.676253  0.802460
251   3.413147 -0.400540  0.183386
......
815   3.214796  1.126532 -1.449776
```

异常值处理

- 将异常值都转换为3或-3

```
In [97]: df[np.abs(df)>3] = np.sign(df)*3

In [98]: df.describe()
Out[98]:
                 0            1            2
count  1000.000000  1000.000000  1000.000000
mean     -0.005124    -0.015942     0.036636
std       1.001422     0.997403     1.033773
min      -2.717053    -3.000000    -3.000000
25%      -0.688916    -0.692900    -0.675153
50%      -0.017609    -0.011644     0.016412
75%       0.669723     0.641029     0.744385
max       3.000000     3.000000     3.000000
```

案例:Boston房价异常值(1)

- 数据集常先标准化,再检测异常值
- Boston房价数据集第4个特征是二元变量(0或1)无需考虑异常值,其他变量都是连续型数值,都需要找出异常值.
- 利用StandardScaler进行标准化,然后找出绝对值大于3的值的位置

案例:Boston房价异常值(2)

```
In [46]: from sklearn import datasets

In [47]: from sklearn.preprocessing import StandardScaler as
   ss

In [48]: ds = datasets.load_boston().data

In [49]: cont_v = [n for n in range(np.shape(ds)[1]) if n!=3]

In [50]: norm_ds = ss().fit_transform(ds[:,cont_v])

In [51]: outlier_row,outlier_col = np.where(np.abs(norm_ds)>3)

In [52]: print list(zip(outlier_row,outlier_col))
[(55, 1), (56, 1), (57, 1), (102, 10),... ]
```

噪声

- 噪声是被测量特征的随机错误,可用数据平滑技术来去除噪声.
- 分桶方法:将数据分配到若干个桶中,然后根据邻近数据来平滑数据
- 分桶可以采用等频数或等宽方法

例:平滑噪声

- **[4, 8, 15, 21, 21, 24, 25, 28, 34]等频数划分为容量为3的桶,然后用桶均值平滑和桶边界平滑**

等频数分桶	桶均值平滑	桶边界平滑
桶1：[4, 8, 15]	桶1：[9, 9, 9]	桶1：[4, 4, 15]
桶2：[21, 21, 24]	桶2：[22, 22, 22]	桶2：[21, 21, 24]
桶3：[25, 28, 34]	桶3：[29, 29, 29]	桶3：[25, 25, 34]

处理重复数据(2)

- 重复数据一般不能为分析模型提供新信息,所以应删除.
- DF利用duplicated来检测重复行, 返回布尔值Series,用于指示每一行是否重复行;而drop_duplicates则用来删除副本(只保留一个副本)

例:识别和删除重复数据

```
In [38]: df = pd.DataFrame({'k1':['one','two']*3+['two'],
    ...:                    'k2':[1,1,2,3,3,4,4]})
```

```
In [39]: df
Out[39]:
    k1  k2
0  one   1
1  two   1
2  one   2
3  two   3
4  one   3
5  two   4
6  two   4
```

```
In [40]: df.duplicated()
Out[40]:
0    False
1    False
2    False
3    False
4    False
5    False
6     True
```

```
In [41]: df.drop_duplicates()
Out[41]:
    k1  k2
0  one   1
1  two   1
2  one   2
3  two   3
4  one   3
5  two   4
```

处理重复数据(2)

- 前面检测重复时考虑所有列的,但也可以指定某些列

```
In [43]: df.drop_duplicates(['k1'])
Out[43]:
    k1  k2
0  one   1
1  two   1
```

- 默认保留第一个观测值,但可以指定参数**keep='last'**来保留最后一个副本

数据转换(1)

- 将数据转换成模型所要求的形式
- 按某个数学函数来转换
  - 例:x和y之间的非线性关系$y = ae^{bx}$转换成y与log(x)之间的线性关系
  - 从原始数据计算汇总数据
  - 数据泛化
  - 标准化,通过改变度量方式将数据转换到特定区间
- 转换的后果:丧失了值的可解释性

数据转换(2)

- **min-max标准化**

$$v' = \frac{v - \min_A}{\max_A - \min_A}(\max'_A - \min'_A) + \min'_A$$

数据转换(3)

- Series提供map,它接受一个函数或字典定义的映射

```
In [50]: s = pd.Series(range(6))
In [51]: s.map(lambda x: 0 if x<3 else 5)
Out[51]:
0    0
1    0
2    0
3    5
4    5
5    5
In [53]: a2e = {0:'z',1:'o',2:'t',3:'t',4:'f',5:'f'}
In [54]: s.map(a2e)
```

数据转换(4)

```
In [61]: s.replace(0,'zero')
Out[61]:
0    zero
1       1
2       2
...
In [62]: s.replace([1,2],'one/two')
Out[62]:
0          0
1    one/two
2    one/two
3          3
...
In [63]: s.replace([3,4,5],['three','four','five'])
```

虚拟变量(1)

- 将类别型变量转换成虚拟变量矩阵
- 特征A有k个类别值,则A可转换成一个k列矩阵,每一列对应一个类别值,且该列上只包含1或0,说明该值在原数据集中的哪一行出现
  - 例如将性别变量(值为M和F)转换成两个虚拟变量M和F:M取值为1的实例就是性别为M的实例,F取值为1的实例就是性别为F的实例

虚拟变量(2)

- pandas的get_dummies函数可以实现这个转换

```
In [85]: df = pd.DataFrame({'no':range(5),
    ...:                    'sex':['F','F','M','M','F']})
In [86]: df
Out[86]:
   no sex
0   0   F
1   1   F
2   2   M
3   3   M
4   4   F
In [87]: pd.get_dummies(df['sex'])
Out[87]:
     F    M
0  1.0  0.0
1  1.0  0.0
2  0.0  1.0
3  0.0  1.0
4  1.0  0.0
```

虚拟变量(3)

- 为虚拟变量名添加前缀,与其他变量的数据合并.

```
In [91]: dummy = pd.get_dummies(df['sex'],prefix='X')

In [93]: df_dummy = df[['no']].join(dummy)

In [94]: df_dummy
Out[94]:
   no  X_F  X_M
0   0  1.0  0.0
1   1  1.0  0.0
2   2  0.0  1.0
3   3  0.0  1.0
4   4  1.0  0.0
```

数据约简

- 数据约简技术可用来得到数据集的约简表示
  - 容量变小但保持了原始数据的大多数信息
  - 使得分析更高效,产生几乎相同的分析结果
- 数据约简技术
  - 聚合
  - 数据离散化
  - 维度约简

数据离散化

- 数据离散化技术将连续型特征的值域划分成区间,然后用区间标签来代替实际数据值,从而减少简化了原始数据.
- 分桶方法也可用作离散化方法.
  - 等宽分桶:各桶数据不均,对异常值敏感
    - 分桶前检测异常值
  - 等频数分桶:导致同类别数据分入不同桶
    - 分桶后调整相邻桶的边界值

离散化:分桶(1)

- pandas函数cut(data,bins)将一维数据集data按照bins的规定进行分桶,其中bins可以是一个整数或一个数值序列.

```
In [64]: ages=[20,22,25,27,21,23,37,31,61,45,41,32]
In [65]: bins = [18,25,35,60,100]
In [66]: cats = pd.cut(ages,bins)
In [67]: cats
Out[67]:
[(18, 25], (18, 25], (18, 25], (25, 35], (18, 25], ..., (25,
   35], (60, 100], (35, 60], (35, 60], (25, 35]]
Length: 12
Categories (4, object): [(18, 25] < (25, 35] < (35, 60] < (60,
   100]]
```

离散化:分桶(2)

- cut返回描述分桶结果的Categorical对象
  - 属性categories可看作是为每个桶命名,例如第一个桶的名字是(18,25].
  - codes属性是原数据所属桶的标记。

```
In [68]: cats.categories
Out[68]: Index([u'(18, 25]', u'(25, 35]', u'(35, 60]', u'(60,
   100]'], dtype='object')

In [69]: cats.codes
Out[69]: array([0, 0, 0, 1, 0, 0, 2, 1, 3, 2, 2, 1],
   dtype=int8)
```

离散化:分桶(3)

- 桶区间是左开右闭的
  - 通过参数**right=False**可建立左闭右开的桶
- **pd.value_counts**可对桶中数据计数

```
In [79]: pd.value_counts(cats)
Out[79]:
(18, 25]     5
(35, 60]     3
(25, 35]     3
(60, 100]    1
```

离散化:分桶(4)

- 可以利用labels参数自定义桶的名称
  - 缺省采用整数标号

```
In [80]: bin_names =
   ['Youth','YoungAdult','MiddleAged','Senior']

In [81]: pd.cut(ages,bins,labels=bin_names)
Out[81]:
[Youth, Youth, Youth, YoungAdult, Youth, ...,
   YoungAdult, Senior, MiddleAged, MiddleAged,
   YoungAdult]
Length: 12
Categories (4, object): [Youth < YoungAdult <
   MiddleAged < Senior]
```

离散化:分桶(5)

- 如果传递桶的个数而不是桶的边界给cut则进行等宽分桶
  - 参数precision表示桶边界保留几位小数

```
In [82]: data = np.random.rand(20)

In [83]: pd.cut(data,4,precision=2)
Out[83]:
[(0.25, 0.48], (0.022, 0.25], (0.7, 0.93], (0.25,
  0.48], (0.25, 0.48], ..., (0.48, 0.7], (0.25, 0.48],
  (0.022, 0.25], (0.25, 0.48], (0.022, 0.25]]
Length: 20
Categories (4, object): [(0.022, 0.25] < (0.25, 0.48] <
  (0.48, 0.7] < (0.7, 0.93]]
```

离散化:分桶(6)

- 另一个分桶方法qcut根据分位数来分桶
  - 下例中的参数4表示对数据集进行四等分

```
In [84]: data = np.random.randn(1000)

In [85]: cats = pd.qcut(data,4)

In [87]: pd.value_counts(cats)
Out[87]:
(0.678, 3.449]      250
(0.0228, 0.678]     250
(-0.603, 0.0228]    250
[-3.281, -0.603]    250
```

离散化:3-4-5规则(1)

- 3-4-5规则可将数值数据分为相对均匀的自然的区间.
- 给定数据划分为3或4或5个相对等宽的区间,由值域的最高有效位确定桶数

离散化:3-4-5规则(2)

- 如果最高有效位覆盖
  - 3,6,7,9个不同值: 则划分为3个区间.
    - 其中3,6,9的情况形成3个等宽区间,7的情况形成宽度为2-3-2的3个区间.
  - 2,4,8个不同值:则划分为4个等宽区间
  - 1,5,10个不同值,则划分为5个等宽区间

例:3-4-5规则(3)

- 对(-35,478)离散化
  - 先不考虑异常值:假设处于5%和95%百分位数之间的数据介于-15和188之间.

1.Min = -35,Max=478.

2.将-15和188舍入到最高有效位(百位), 可得-100和200

3.(200-(-100))/3=3,即最高有效位(百位)覆盖三种值,所以5%到95%百分位之间的数据划分为3个等宽区间(-100,0], (0,100], (100,200]

例:3-4-5规则

4.异常值处理:由于(-100,0]覆盖了Min,所以可将此区间的左边界进行调整,以使区间范围尽量小.

由于Min=-35的最高位为十位,将Min舍入到十位可得-40,于是(-100,0]可调整为(-40,0],而(100,200]不包含Max=478,需要建立一个新区间.将Max舍入到最高有效位可得500,则新区间为(200,500]

数据约简

- 数据约简技术可用来得到数据集的约简表示
  - 容量变小但保持了原始数据的大多数信息
  - 分析更高效,分析结果几乎相同
- 例如:聚合也可看作是数据约简技术
  - 每天的销售汇总成按月的销售量
- 最常用的数据约简是维度约简

维度约简

- 减少数据集的维度即特征数
  - 去除无关或弱相关的,冗余的特征
- 例如:前述虚拟变量导致的两列M和F
  - M为1时F必为0,M为0时F必定为1
  - 将M和F之一去掉没有任何信息损失

特征子集选择

- n个特征有2^n个子集,穷尽搜索不可行
- 启发式方法:一般都是贪婪算法,即在搜索特征空间时总是作出当前的最佳选择,并期待局部最优选择会带来全局最优解
  - 逐步向前选择:从空特征集出发逐步加入最佳特征
  - 逐步向后删除:从全体特征集出发逐步删除最差特征
  - 向前选择与向后删除结合

主成分分析

- **PCA:找出能反映最大偏差(能量)的特征线性组合,称为主成分,构成新特征空间**

  $$PC_j = (a_{j1} \times A_1) + (a_{j2} \times A_2) + \ldots + (a_{jn} \times A_n)$$

  - 第一个主成分定义为所有特征线性组合中具有最大偏差的线性组合
  - 后续主成分定义为反映最大剩余偏差的特征线性组合,并且与前面的主成员无关
- 原数据集被变换到特征数更少的新空间

PCA过程

- 各属性零均值化(即减去属性均值)
- 求协方差矩阵
- 求协方差矩阵的特征值及对应特征向量
- 对特征值从大到小排序
- 将特征向量按对应特征值大小从上到下按行排列成矩阵
- 将数据转换到k个特征向量构建的新空间中

案例:PCA

- 另附文档

End